

**WHAT IS CLAIMED IS:**

1           1.       An execution unit for execution of multiple context threads comprises:  
2                    an arithmetic logic unit to process data for executing threads;  
3                    control logic to control the operation of the arithmetic logic unit;  
4                    a general purpose register set to store and obtain operands for the arithmetic  
5 logic unit, the register set constructed with a two-ported random access memory, with the  
6 register set divided into a plurality of banks and with two different words from each bank of  
7 the register set can be read and written in the same processor cycle.

1           2.       The execution unit of claim 1 wherein the register set is logically partitioned  
2 into a plurality of relatively addressable windows.

1           3.       The execution unit of claim 2 wherein the number of windows of the register  
2 set is according to the number of threads that can execute in the processor.

1           4.       The execution unit of claim 1 where the relative addressing allows the  
2 currently executing thread to access to any of the registers relative to the starting point of a  
3 window of registers.

1           5.       The execution unit of claim 1 wherein the register set is absolutely  
2 addressable where any one of the addressable registers may be accessed by the currently  
3 executing thread by providing the exact address of the register.

1           6.       The execution unit of claim 1 wherein the control logic further comprises:  
2 context event switching logic fed by signals from a plurality of shared resources with the  
3 signals causing the context event logic to indicate that threads are either available or  
4 unavailable for execution.

1           7.       The execution unit of claim 6 wherein the control logic addresses a set of  
2 memory locations for storing a list of available threads that correspond to threads that are  
3 ready to be executed and a set of memory locations for storing a list of unavailable threads  
4 that are not ready to be executed.

1           8.       The execution unit of claim 7 wherein execution of a context swap instruction  
2 causes a currently running thread to be swapped out to the unavailable thread memory set  
3 and a thread from the available thread memory set to begin execution within a single  
4 execution cycle.

1           9.       The execution unit of claim 8 wherein execution of the context swap  
2 instruction specifies one of the signal inputs and upon receipt of the specified signal input  
3 causes the swapped out thread to be stored in the available thread memory set.

1           10.      The execution unit of claim 8 wherein execution of the context swap  
2 instruction specifies a sequence number change and upon receipt of the specified sequence  
3 number change causes the swapped out thread to be stored in the unavailable memory set.

1           11.      The execution unit of claim 8 wherein execution of the context swap  
2 instruction specifies an inter-thread signal input and upon receipt of the specified inter-  
3 thread signal causes the swapped out thread to be stored in the available memory set.

1           12.      The execution unit of claim 8 wherein execution of the context swap  
2 instruction specifies a voluntary swap operation and causes a context swap if there is a  
3 thread in the available memory set ready to be executed.

1           13.      The execution unit of claim 8 wherein execution of the context swap  
2 instruction specifies a defer\_one operation which causes execution of one more instruction  
3 and then causes the current context to be swapped out.

1           14.      The execution unit of claim 8 wherein the context event switching logic  
2 further comprises:  
3               a ctx\_enable register and with execution of the context swap instruction  
4 specifying a kill operation causing the ctx\_enable bit to be set to indicate that this thread is  
5 not available for execution until the bit is cleared by another instruction.

1           15.    A method for executing multiple context threads comprises:  
2                   processing data for executing threads within an arithmetic logic unit;  
3                   operating control logic to control the arithmetic logic unit;  
4                   storing and obtaining operands for the arithmetic logic unit within a general  
5   purpose register with the register set constructed with a two-ported random access memory;  
6   and  
7                   arranging the register set into a plurality of banks where two separate words  
8   from separate banks of the register set can be read and written in the same processor cycle.

1           16.    The method of claim 15 wherein the register is relatively addressable.

1           17.    The method of claim 15 further comprising:  
2                   arranging the register set into a number of windows according to the number  
3   of threads that can execute in the processor.

1           18.    The method of claim 17 wherein storing and obtaining further comprises:  
2                   addressing the registers by the currently executing thread by providing a  
3   relative register address that is relative to the starting point of a window of registers.

1           19.    A processor unit comprises:  
2                   an execution unit for execution of multiple context threads comprising:  
3                   an arithmetic logic unit to process data for executing threads;  
4                   control logic to control the operation of the arithmetic logic unit;  
5                   a general purpose register set to store and obtain operands for the arithmetic  
6   logic unit, the register set constructed with a two-ported random access memory.

1           20.    The processor of claim 19 wherein the register set is logically partitioned into  
2   a plurality of relatively addressable windows where the number of windows of the register  
3   set is according to the number of threads that can execute in the processor.

1           21.     The processor of claim 20 where the relative addressing allows the currently  
2     executing thread to access to any of the registers relative to the starting point of a window of  
3     registers.

1           22     The processor of claim 20 wherein the register set is absolutely addressable  
2     where any one of the addressable registers may be accessed by the currently executing thread  
3     by providing the exact address of the register.

1           23.     The processor of claim 20 further comprises:  
2                   a set of memory locations for storing a list of available threads that correspond  
3     to threads that are ready to be executed;  
4                   a set of memory locations for storing a list of unavailable threads that are not  
5     ready to be executed; and  
6                   context event switching logic fed by signals from a plurality of shared  
7     resources with the signals causing the context event logic to indicate which threads are either  
8     available or unavailable for execution.

1           24     The processor of claim 23 wherein execution of a context swap instruction  
2     causes a currently running thread to be swapped out to the unavailable thread memory set  
3     and a thread from the available thread memory set to begin execution within a single  
4     execution cycle.

1           25.     The processor of claim 23 wherein execution of a context swap instruction  
2     specifies one of the signal inputs and upon receipt of the specified signal input causes the  
3     swapped out thread to be stored in the available thread memory set.

1           26.     The processor of claim 23 wherein execution of a context swap instruction  
2     specifies a defer\_one operation which causes execution of one more instruction and then  
3     causes the current context to be swapped out.

1           27.    The processor of claim 23 wherein the context event switching logic further  
2 comprises:

3                   a ctx\_enable register and with execution of a context swap instruction  
4 specifying a kill operation causing the ctx\_enable bit to be set to indicate that this thread is  
5 not available for execution until the bit is cleared by another instruction.

1           28.    A computer program product residing on a computer readable medium for  
2 causing a processor to perform a function comprises instructions causing the processor to:

3                   perform a context swapping operation to cause a currently running thread to  
4 be swapped out to an unavailable thread memory set and a thread from an available thread  
5 memory set to begin execution within a single execution cycle.

1           29.    The product of claim 28 wherein a context swapping operation specifies a signal  
2 input and upon receipt of the specified signal input causes the swapped out thread to be  
3 stored in the available thread memory set.    .